VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SYSTÉM PRO KONTROLU SLOVNÍKŮ

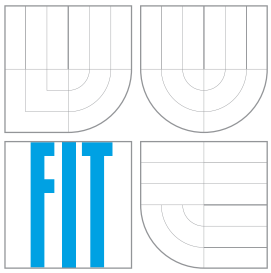
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR SOLANSKÝ

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SYSTÉM PRO KONTROLU SLOVNÍKŮ

DICTIONARY CHECKER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR SOLANSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2014

Abstrakt

Práce je zaměřena na implementaci informačního systému pro kontrolu a opravu překladových a výkladových elektronických slovníků ve formátu LMF. Systém nabízí sedm typů kontrol a jednu opravu hromadně měnící obsahy slovníků. V technické zprávě jsou popsány nejdůležitější použité technologie, konceptuální návrh systému i kontrol samotných, důležité implementační prvky a výsledky se statistikou tohoto informačního systému.

Abstract

The work is focused on implementation of information system for checking and repairing electronic dictionaries in LMF format. This system offers seven types of controls and one repair, mass changing content of dictionaries. The documentation describes the most important used technologies, conceptual model of the system, interesting implementation details and results with statistics of this information system.

Klíčová slova

elektronické slovníky, kontrola, Lexical Markup Framework, Extensible Markup Language, Relax NG, Python, Shell, GNU Aspell, libma

Keywords

electronic dictionaries, check, Lexical Markup Framework, Extensible Markup Language, Relax NG, Python, Shell, GNU Aspell, libma

Citace

Petr Solanský: Systém pro kontrolu slovníků, bakalářská práce, Brno, FIT VUT v Brně, 2014

Systém pro kontrolu slovníků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Solanský
19. května 2014

Poděkování

Rád bych poděkoval panu Doc. RNDr. Pavlu Smržovi, Ph.D. za odbornou pomoc a vedení mé bakalářské práce.

Obsah

1 Úvod **3**

2 Rozbor použitých metod **4**
2.1 XML . . . 4
2.1.1 Syntaxe . . . 4
2.1.2 Zpracování XML . . . 6
2.1.3 Validace . . . 7
2.2 Lexical Markup Framework . . . 9
2.2.1 Třídy v LMF . . . 10
2.3 GNU Aspell . . . 12
2.4 Morfologický analyzátor pro češtinu . . . 12
2.5 Python . . . 12
2.6 Shell . . . 12

3 Návrh a externí komponenty **13**
3.1 Model systému . . . 13
3.2 Vstupní data . . . 13
3.3 Kontrola formátu LMF . . . 13
3.3.1 Relax NG . . . 14
3.4 Srovnání se slovníkem pro kontrolu pravopisu . . . 14
3.4.1 GNU Aspell . . . 14
3.4.2 Libma . . . 14
3.5 Atributy slovníků . . . 14
3.6 Abecední uspořádání . . . 15
3.7 Zpětný překlad . . . 15
3.8 Četnost hesel . . . 16

4 Implementace systému **17**
4.1 Jádro systému . . . 17
4.2 Zpracování vstupních dat . . . 18
4.3 Kontrola formátu LMF . . . 18
4.4 Srovnání se slovníkem pro kontrolu pravopisu . . . 18
4.4.1 GNU Aspell . . . 18
4.4.2 Libma . . . 19
4.5 Atributy slovníků . . . 19
4.6 Abecední uspořádání . . . 21
4.7 Zpětný překlad . . . 22
4.8 Četnost hesel . . . 22

5 Statistiky a výsledky **23**
5.1 Libma 23
5.2 GNU Aspell 24
5.3 Atributy slovníků 25
5.4 Abecední uspořádání 26
5.5 Zpětný překlad 27
5.6 Souhrnná statistika 29

6 Závěr **30**

A Obsah CD **33**

Kapitola 1

Úvod

Elektronické slovníky jsou jedním z nejvyužívanějších zdrojů informací při výuce cizích jazyků, široké uplatnění však mají také v mezinárodně obchodujících firmách i mezi turisty. V dnešní době chytrých telefonů a tabletů jsou tištěné slovníky zcela nahrazeny slovníky elektronickými. Ty předčí tištěné slovníky zejména praktičností a efektivnějším vyhledáváním. Z důvodu celosvětového využívání této technologie je potřeba klást důraz na jejich vývoj a správnost.

Při vývoji a generování těchto slovníků dochází mnohdy k chybám, které je potřeba v co největší míře odstranit. Tématem této bakalářské práce je systém pro kontrolu překladových a výkladových slovníků, který se touto problematikou zabývá. Cílem takového informačního systému je odhalení jak triviálních, tak komplikovanějších chyb, jejich automatická oprava nebo popřípadě návrh na provedení změn.

Systém je implementován a testován pomocí elektronických slovníků vyvíjených na Fakultě informačních technologií Vysokého učení technického v Brně ve Výzkumné skupině znalostních technologií. Práce obsahuje část praktickou a teoretickou. Praktickou částí je u této práce sada skriptů v programovacích jazycích Python a Shell tvořící výsledný informační systém. Teoretickou částí je technická zpráva k tomuto systému, obsahující úvod do použitých technologických metod, koncepci systému, detaily implementace a výsledky se statistikou tohoto informačního systému.

Kapitola 2

Rozbor použitých metod

V této kapitole jsou stručně popsány základní technologie. A nejen ty, které byly v této práci použity, ale také ty, ze kterých byly implementované technologie vybrány. Slovníky, pro které je tento systém vyvíjen, jsou psány v jazyce XML (viz kapitola 2.1) a dodržují formát LMF (viz kapitola 2.2). Z toho důvodu se tato kapitola zabývá z většiny technologiemi řešící tento problém.

2.1 XML

XML [18] (z angl. *Extensible Markup Language*) je značkovací jazyk, který je standardem schváleným konsorciem W3C [1]. Definuje formát dokumentů a obecnou syntaxi. Formát tohoto jazyka je flexibilní, umožňuje úpravy do mnoha odvětví informačních technologií. Využití je například v oblasti webových stránek, elektronické výměny dat, vektorové grafiky, vzdálené volání procedur, serializace objektů a dalších. U jazyka XML je také dostupné velké množství knihoven v nejrůznějších programovacích jazycích, které umí z formátu XML data číst a také do něj zapisovat.

XML je následníkem značkovacího jazyka SGML [19] (z angl. *Standard Generalized Markup Language*), ze kterého vychází například celosvětově známý jazyk HTML [19] (z angl. *HyperText Markup Language*), využívaný k publikaci dokumentů na internetu (viz obrázek 2.1). Na rozdíl od XML jazyk HTML obsahuje informace o způsobu zobrazení dat.

XML lze nazvat metajazykem, díky kterému lze vytvářet vlastní jazyky. Neobsahuje konkrétní značky a tak si může každý vymyslet značky vlastní.

Příklad jednoduchého XML:

```
<osoba>
    <jmeno>Jaroslav</jmeno>
    <prijmeni>Kudrna</prijmeni>
    <vek>46</vek>
</osoba>
```

2.1.1 Syntaxe

XML je meta-značkovací jazyk, tudíž nedefinuje pevně danou množinu elementů a značek. To znamená, že uživatel má možnost definovat si své vlastní elementy a značky tak, aby vyhovovaly právě jeho potřebám. Na druhou stranu je striktně dána gramatika dokumentů, která specifikuje umístění značek, elementů a atributů přiřazených těmto elementům.

Obrázek 2.1: Vztahy mezi uvedenými jazyky. Zdroj: [18]

Díky této gramatice je možné využívat analyzátory, které nám usnadňují čtení dat z XML formátu, ale také usnadňují zápis dat do XML souborů.

Základní prvky jazyka XML:

- **Prolog** – Prolog, jinak také XML deklarace, je element obsahující atributy, definující verzi a kódování daného XML souboru. Například:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  ```

- **Elementy** – Element je základní stavební kámen každého XML souboru. Existuje takzvaný kořenový element, který obsahuje celou strukturu dokumentu, tedy mimo počáteční prolog. Elementy jsou vyznačeny pomocí tagů. Ve většině případů jeden element odpovídá dvěma tagům – počátečnímu a koncovému. Element může být zapsán také jako prázdný za pomocí pouze jednoho tagu. Obsahem elementu kromě textu může být jeden nebo více elementů podřazených. Díky této vlastnosti lze tvořit komplexnější XML struktury. Příklad párového a prázdného elementu:

  ```
  <jmeno>Jaroslav</jmeno>

  <osoba vek="46"/>
  ```

- **Atributy** – Přidání vlastností, doplnění informací nebo upřesnění významu elementu lze dosáhnout přidáním atributu. Tento prvek lze umístit každému počátečnímu tagu elementu. Jeden element smí zahrnovat více než jeden atribut. Příklad atributů `jmeno` a `prijmeni` u prázdného elementu `osoba`:

  ```
  <osoba jmeno=„Karel“ prijmeni=„Novotný“/>
  ```

- **Znakové entity** – Znaky jako `<`, `>` nebo `¨` jsou využity při zápisu zdrojového kódu u elementů a atributů. Není tak možné je zapsat do obsahů elementů a atributů přímo. Zde je potřeba využít znakových entit. V tabulce 2.1.1 jsou u znaků uvedeny entity, pomocí kterých je možné znaky do textu zapsat.

- **CDATA** – V případě častého využití znakových entit může být psaní textu poněkud nepohodlné. Sekce CDATA umožňuje znaky zapisovat přímo. Využití této sekce je vhodné zejména v případech sázení programovacího kódu. Příklad zápisu:

& (ampersand)	&amp
> (větší než)	&gt
< (menší než)	&lt
“ (uvozovky)	&quot
’ (apostrof)	&apos

Tabulka 2.1: Tabulka znakových entit.

```
<script language="JavaScript">
  <![CDATA[
     if (a~> 2 && b > 4)
      {
       document.writeln("<p>Ahoj</p>");
      }
    ]]>
</script>
```

- **Komentáře** – Jak je zvykem, každý počítačový jazyk umožňuje do svého zdrojového kódu zapisovat komentáře. Syntaxe je u formátu XML stejná jak pro komentář řádkový, tak blokový. Text zakomentujeme pomocí červeně zvýrazněných značek:

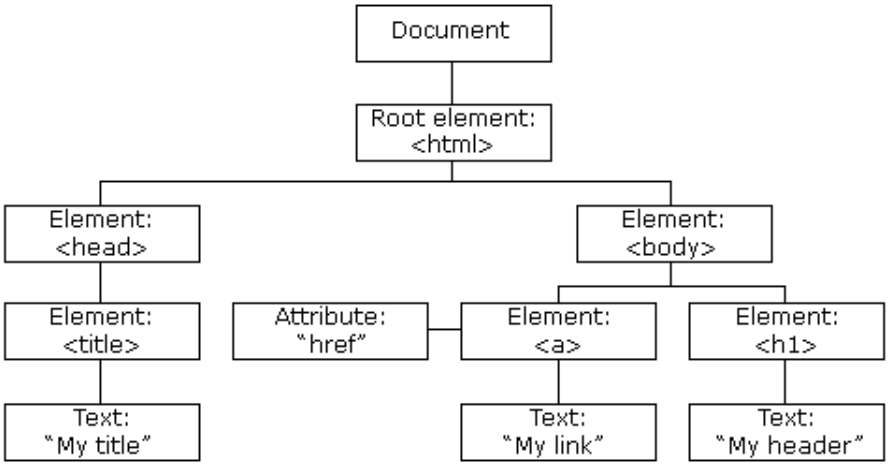
```
<osoba>
  <!-- <jmeno>Jaroslav</jmeno>
       <prijmeni>Kudrna</prijmeni> -->
       <vek>46</vek>
</osoba>
```

2.1.2 Zpracování XML

Jazyk XML je pasivní formát využívající se k uchování dat. K efektivní práci s tímto formátem je potřeba využít algoritmy, které nám usnadní čtení dat a generování XML. Dnes existuje mnoho analyzátorů, většina z nich je postavena na jednom ze dvou modelů API. A to objektový model DOM schválený konsorciem W3C a jednoduchý model SAX. Každý z těchto dvou modelů přistupuje ke zpracování XML formátu jiným způsobem, a to dává vývojářům možnost zvolit si nástroj vyhovující konkrétní práci. V programovacím jazyce Python je možné ke zpracování XML využít modul The ElementTree XML API.

DOM

Tento objektový model [6] definuje hierarchii objektů, pomocí nichž popíše každý XML dokument jako rekurzivní seznam seznamů (viz obrázek 2.2). Nejdůležitější vlastností tohoto přístupu je, že potřebuje nejprve celý soubor analyzovat a načíst do paměti, aby následně bylo možné uvedenou datovou strukturu vytvořit. Tato vlastnost se jeví jako problémová při zpracování rozsáhlých souborů a také u souborů, které ve chvíli zpracování modelem DOM nejsou celé k dispozici. Znamená to tedy, že tento model je vhodný zejména pro menší dokumenty a pro aplikace vyžadující náhodný a opakující se přístup k prvkům zpracované XML struktury.



Obrázek 2.2: Stromová struktura modelu DOM. Zdroj: [6]

SAX

Cílem událostmi řízeného modelu SAX [12] je nabídnout alternativu k objektovému modelu DOM. Je tedy zřejmé, že tento model nevytváří v paměti úplný obraz dokumentu, ale aplikuje techniku zpětného volání. To znamená, že u každé syntaktické konstrukce, jako například element či atribut, je vyvolána událost. Tento analyzátor tudíž nemá potřebu téměř žádné paměti a lze ho tedy využít i u rozsáhlých dokumentů.

ElementTree

Modul ElementTree [5] je podobný objektovému modelu DOM, je však jednodušší, protože stromovou strukturu XML představuje jako struktury datových typů jazyka Python `dictionary`[1] a `list`[2]. Z toho důvodu je tento modul ke zpracování XML rychlejší a potřebuje mnohem méně paměti než modul DOM. Moduly SAX a DOM mají ale tu výhodu, že jsou kompatibilní se standardními API, takže uživatelé, kteří jsou již s problematikou obeznámeni, mohou tyto moduly využít bez učení nových technologií.

2.1.3 Validace

Neopomenutelnou vlastností jazyka XML je možnost definovat strukturu dat. Usnadní nám zpracování jak vstupního, tak výstupního zdrojového kódu. Na základě vlastního schématu pro kontrolu validity XML struktury lze snadno určit, zda jsou vstupní data pro naše potřeby validní. Existuje několik způsobů, tedy jazyků pro validaci, jak lze strukturu XML definovat. V některých lze definovat pouze základní prvky XML struktury, jako jsou elementy a atributy, u jiných například datové typy obsahů těchto prvků. Mimo vlastní definice schémat musí navíc každá XML struktura splňovat podmínky takzvaného well-formed dokumentu.

[1] Viz `http://www.tutorialspoint.com/python/python_dictionary.htm`
[2] Viz `http://www.tutorialspoint.com/python/python_lists.htm`

Well-formed

Pokud chceme XML dokument prohlásit za well-formed, musí splňovat následující podmínky:

- Dokument úvodem obsahuje prolog (XML deklaraci).
- Dokument obsahuje kořenový element.
- Dokument obsahuje pouze párové tagy s výjimkou prázdného elementu.
- Obsah atributů je uzavřen do uvozovek nebo apostrofů.
- Jména elementů a atributů jsou case-sensitive.

DTD

DTD [7] (z angl. *Document Type Definition*) je poměrně stará a v mnoha ohledech nedostačující technologie pro definici XML struktury. Největší nevýhodou DTD je neschopnost využití jmenných prostorů. Přesto je stále využívána díky uplatnění v nejrůznějších analyzátorech a aplikacích. DTD popisuje vzájemné uspořádání a zanořování prvků (elementy, atributy). Definuje atributy elementů a obsahy těchto prvků. Zde je však omezení, neumožňuje definovat restrikce jako datum, čas, měna atd. Nevýhodou se může zdát také fakt, že zápis tohoto schématu není XML struktura. Příklad jednoduchého DTD schéma:

```
<!DOCTYPE osoba
[
<!ELEMENT osoba (jmeno,prijmeni,vek)>
<!ELEMENT jmeno (#PCDATA)>
<!ELEMENT prijmeni (#PCDATA)>
<!ELEMENT vek (#PCDATA)>
]>
```

Relax NG

Technologie Relax NG [4] je založena na vzorech, nikoli na datových typech. Validační schéma je XML struktura:

```
<element name="osoba">
  <oneOrMore>
    <element name="jmeno">
      <text/>
    </element>
  <oneOrMore>
  <element name="prijmeni">
     <text/>
  </element>
  <element name="vek">
    <attribute name="name">
      <text/>
    </attribute>
    <text/>
  </element>
</element>
```

Schematron

Technologie DTD a Relax NG, zmíněné v předešlých kapitolách, definují víceméně gramatiku XML struktury. Schematron [16]je však založen na odlišném principu. Tato technologie umožňuje vytvořit tvrzení o přítomnosti či absenci daných vzorů v XML struktuře. Takový vzor lze vytvořit pomocí jazyku XPath [8]. Například:

```
<schema xmlns="http://www.ascc.net/xml/schematron" >
 <pattern name="Hlavni">
  <rule context="osoba">
    <assert test="jmeno">Chybí element "jmeno".</assert>
    <report test="jmeno">Element "jmeno" je v˜pořádku.</report>
    <assert test="prijmeni">Chybí element "prijmeni".</assert>
    <assert test="vek">Chybí element "vek".</assert>
  </rule>
 </pattern>
</schema>
```

XML Schema

Protože tato technologie má ve jméně obecné označení jazyka, tedy XML, používá se také označení XSD [9]. XSD je alternativa k XML schématům jako je DTD. Definuje pravidla pro strukturu XML dokumentu, rozšiřuje vlastnosti DTD o jmenné prostory, datové typy, ale hlavně jeho zápis je v XML. Jednoduchý příklad:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="osoba">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="jmeno" type="xs:string"/>
    <xs:element name="prijmeni" type="xs:string"/>
    <xs:element name="vek" type="xs:decimal"/>
    <xs:element name="narozen" type="xs:date"/>
    </xs:sequence>
   <xs:attribute name="id" type="xs:integer"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

2.2 Lexical Markup Framework

LMF [15] (z angl. Lexical Markup Framework) je ISO standardem schváleným organizací International Organization for Standardization pro zpracování přirozeného jazyka NLP (z angl. natural language processing) a strojově čitelné slovníky, tedy MRD (z angl. machine-readable dictionary).

LMF nám nabízí pravidla zápisu datových struktur, díky kterých lze jak data pro výkladové, tak překladové elektronické slovníky ukládat. LMF popisuje strukturu syntaktickou, sémantickou i morfologickou. Tudíž je tento standard velmi flexibilní, při jeho dodržení je možné slovníky snadno spojovat a kombinovat. Krátký příklad zápisu dat do LMF formátu,

který obsahuje prostředí pro slovníky česko-anglické s jedním slovníkem a jedním heslem. Heslo `auto` obsahuje překlad `car`, další formu zápisu `automobil`, slovní druh v angličtině `commonNoun` a gramatickou informaci opět v angličtině, v tomto případě je to `singular`:

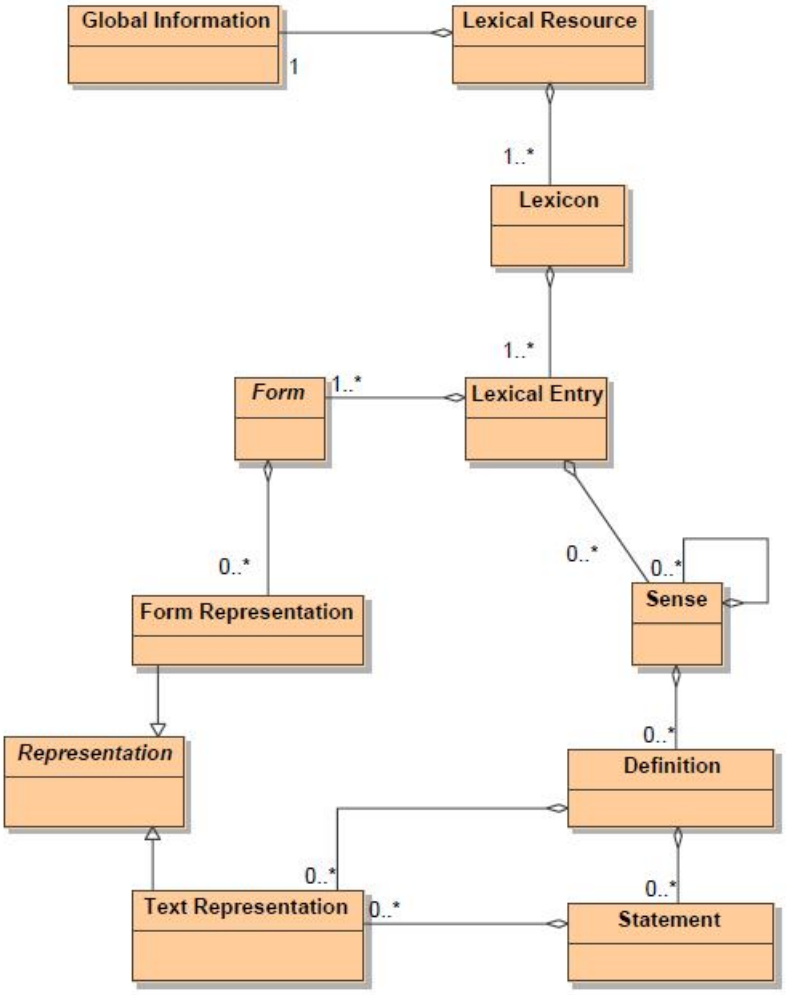
```
<LexicalResource dtdVersion="16">
 <GlobalInformation>
  <feat att="languageCoding" val="UTF-8"/>
  <feat att="fromLanguage" val="cz"/>
  <feat att="toLanguage" val="en"/>
 </GlobalInformation>
 <Lexicon>
  <LexicalEntry id="00000001">
    <feat att="partOfSpeech" val="commonNoun"/>
    <Lemma>
      <feat att="writtenForm" val="auto"/>
    </Lemma>
    <WordForm>
       <feat att="writtenForm" val="automobil"/>
       <feat att="grammaticalNumber" val="singular"/>
    </WordForm>
    <Sense>
      <Equivalent>
         <feat att="writtenForm" val="car"/>
      </Equivalent>
    </Sense>
  </LexicalEntry>
 </Lexicon>
</LexicalResource>
```

2.2.1 Třídy v LMF

Veškerá data uložená pomocí formátu LMF jsou rozdělena do tříd definovaných ISO standardem. Pro jednoznačné určení významu uložených dat jsou třídy ve vztazích (viz obrázek 2.3).

Popis vlastností jednotlivých tříd:

- **Lexical Resources** – třída reprezentující kořenový element, obsahuje veškeré použité slovníky.
- **Global Information** – třída obsahující základní informace dokumentu. Jako minimum musí tato třída obsahovat informaci o použité normě kódování.
- **Lexicon** – reprezentace jednoho slovníku. Minimum u této třídy je alespoň jedno obsažené heslo, tedy třída `Lexical Entry`.
- **Lexical Entry** – reprezentuje jedno heslo ve slovníku. Minimem je obsažená třída `Lemma`.
- **Lemma** – v této třídě jsou uvedeny základní informace třídy `Lexical Entry`.
- **Word Form** – uvádí další tvary původního hesla a gramatické vlastnosti, jako například výslovnost. Pokud by u jednoho hesla bylo zapotřebí uvést více vlastností stejného typu, využije se třída `Form Representation`.



Obrázek 2.3: Základní třídy LMF a jejich vztahy. Zdroj: [15]

- **Form Representation** – představuje jednu z variant určité vlastnosti.
- **Sense** – třída obsahuje překlady, či významy původního hesla. Obsahuje například třídu `Equivalent`.
- **Definition** – popisuje význam hesla, tedy třídy `Lexical Entry`.
- **Text Representation** – pokud je zapotřebí poskytnout informaci ve více jazycích, využije se této třídy. Obsažena je informace použitého jazyka.
- **Equivalent** – překlad hesla, s uvedeným jazykem překladu.
- **Context** – třída uvádí jeden nebo více příkladů použití původního hesla.
- **Subject Field** – specifikace domény, ze které význam pochází.

2.3 GNU Aspell

GNU Aspell [10] je open source program kontrolující pravopis slov. Byl vyvinut jako náhrada za program Ispell. Na rozdíl od něj Aspell umožňuje kontrolovat také dokumenty v kódování UTF-8 bez použití speciálních slovníků. Druhou nejvýznamnější výhodou je, že program Aspell dokáže při své kontrole uplatnit více než jeden slovník. Dnes máme k dispozici slovníky přibližně v sedmdesáti světových jazycích. Hlavním správcem programu GNU Aspell je Kevin Atkinson [13].

2.4 Morfologický analyzátor pro češtinu

Morfologický slovník a morfologický analyzátor pro češtinu [17] nám umožňuje použít knihovnu `pylibma` obsahující rozhraní knihovny `libma` pro práci v programovacím jazyce Python (viz 2.5). Díky tomuto rozhraní lze podobně jako programem GNU Aspell (viz 2.3) kontrolovat pravopis slov, v tomto případě pouze českých.

2.5 Python

Python [2] je open source objektově orientovaný programovací jazyk. Tento jazyk bývá také označován jako skriptovací, i když možnosti využití jsou zde daleko širší. Už od roku 1991, kdy přišel na svět, byl navrhnut tak, aby zprostředkovával nástroje pro vývoj plnohodnotných aplikací včetně uživatelského grafického prostředí. Python lze také nazvat programovacím jazykem hybridním a to proto, že umožňuje při vývoji aplikací uplatnit paradigma procedurální, objektově orientované, ale z části také paradigma funkcionální. Díky této široké škále vlastností lze Python uplatnit při vývoji aplikací nejrůznějšího typu. Tento jazyk je velmi oblíbeným, důvodů může být hned několik, například podpora jmenných prostorů, využití výjimek, mnoho dostupných knihoven a modulů pro usnadnění implementace a značná podpora práce s textem.

2.6 Shell

Shell [3] je označení pro program tvořící uživatelské rozhraní, který uživateli umožní využít funkce jádra operačního systému jako spouštění programů, práce s jejich vstupy a výstupy. Shellový skript je v unixových operačních systémech textový soubor, který obsahuje příkazy Shellu. Takové skripty jsou užitečné při řízení více programů a práci s jejich vstupy a výstupy.

Kapitola 3

Návrh a externí komponenty

V této kapitole je popsán návrh systému pro kontrolu překladových a výkladových elektronických slovníků a také kontroly a operace samotné. Dále jsou zde zmíněny externí komponenty, které usnadňují, ale především zefektivňují vybrané implementované kontroly.

3.1 Model systému

Implementované kontroly a opravy působí navenek jako jeden celistvý systém. Na vstup se přivádí překladový nebo výkladový slovník, parametr určující prováděnou operaci, tedy typ opravy či kontroly, a v některých případech také jazyk vstupního slovníku. K dispozici je dále `makefile`, který nám umožní zpracovat všechny slovníky v zadaném adresáři. Tento `makefile` spustí pro každý slovník veškeré dostupné kontroly. Výstupem systému jsou vygenerované soubory, obsahující buďto vypsané chyby slovníků, provedené změny, nebo návrhy na změny. Ke každé operaci je také vygenerován soubor statistik s uvedeným počtem kontrolovaných a chybných hesel vstupního slovníku, procentuální vyjádření těchto statistik a celkový čas kontroly. Všechny generované soubory jsou uloženy do příslušných adresářů.

3.2 Vstupní data

Jak již bylo zmíněno, vstupními daty tohoto systému jsou elektronické slovníky ve formátu LMF. Jeden překladový slovník v tomto formátu je složen ze dvou souborů. Například u slovníku česko-anglického anglicko-českého první soubor obsahuje data, kde jsou česká slova přeložena do angličtiny a druhý soubor anglická slova přeložená do češtiny. Výkladovým slovníkem je tedy pouze jeden soubor. Průměrná velikost jednoho takového souboru obsahujícího slovníková data ve formátu LMF je okolo 100 MB. Proto by při implementaci měl být kladen důraz na zpracování vstupních dat. Při neefektivním zpracování by systém u rozsáhlého párového slovníku mohl být prakticky nepoužitelný.

3.3 Kontrola formátu LMF

Chyby při generování elektronických slovníků mohou způsobovat nejen hesla nesplňující státní normu daného jazyka, ale také nevalidní elementy či atributy, které formát LMF nedodržují.

Než se nad vstupními slovníky začnou provádět jakékoliv operace, měla by být nejprve prověřena validita jejich LMF formátu. Při nevalidních zpracování slovníků by mohlo docházet k chybám systému. A v případě operace, kdy systém generuje opravený slovník, přejímá LMF formát ze slovníku vstupního. Opravený vygenerovaný slovník by tedy byl s velkou pravděpodobností, co se týče LMF formátu, také nevalidní. O tento problém se v systému stará technologie Relax NG.

3.3.1 Relax NG

Tato technologie umožňuje vytvořit schéma k validaci vstupních elektronických slovníků. Důvodem volby Relaxu NG oproti DTD a Schematronu je zejména to, že schéma jeho zápisu je v jazyce XML. DTD a Schematron ve svých vzorech využívají také datové typy, kterými omezují hodnoty elementů a atributů. A tuto vlastnost systém k validaci vstupních slovníků zcela nevyužije. Zápis schématu v XML využívá také XSD, ale protože Relax NG využívá k validaci vzory, byl pro tuto práci vybrán jako vhodnější varianta.

3.4 Srovnání se slovníkem pro kontrolu pravopisu

Kontrolovat pravopis hesel je bezpochyby nutností, důvodů může být hned několik. Při generování elektronických slovníků může docházet k chybám, je tedy pochopitelné, že některá hesla mohou být vygenerována nekorektně. Dále by slovníky neměly hledané heslo přeložit na výrazy nespisovné. Neposlední důvod je z pohledu uživatele pravděpodobně zanedbatelný, ale vývojář elektronických slovníků by tuto vlastnost neměl opomenout. Pokud jsou ve slovnících nespisovná hesla, je velice pravděpodobné že uživatel takové heslo nikdy vyhledávat nebude. Uložená data u těchto hesel, jako jsou překlady, příklady použití, gramatické informace a další, jsou tedy ve slovnících zcela zbytečná.

Testování pravopisné korektnosti hesel je v tomto systému rozděleno do dvou částí, kontrola českých slovníku a kontrola slovníků v cizích jazycích. U obou metod je využito ověřených externích rozhraní vyvíjených právě pro tuto problematiku.

3.4.1 GNU Aspell

Pro kontrolu cizích jazyků jsou na školním serveru `minerva1` dostupné programy GNU Aspell a Ispell. Vybrán byl program GNU Aspel, zejména z důvodů uvedených v kapitole 2.3. Tento program je využíván pro kontrolu cizího jazyka anglického, německého, francouzského, ruského a slovenského.

3.4.2 Libma

V případě jazyka českého bylo využito programu neveřejného. Tím je morfologický analyzátor, vyvíjený na Fakultě informačních technologií Vysokého učení technického v Brně ve Výzkumné skupině znalostních technologií. Konkrétně knihovny `libma`, de facto `pylibma`, která zprostředkovává rozhraní pro jazyk Python.

3.5 Atributy slovníků

Formát LMF obsahuje mnoho tříd, do kterých lze o jednom konkrétním hesle uložit nespočet informací. Tyto třídy jsou specifikovány předem danými jmény elementů a atributů. Každá

taková třída potom nabývá určité hodnoty.

Systém má předem daný seznam povolených hodnot. Na základě tohoto seznamu systém slovníky automaticky opravuje, nebo navrhuje atributy, které by měly být změněny. Samozřejmostí jsou generované statistiky. Tyto atributy mohou upřesnit význam nalezených překladů, nebo omezit množinu hodnot vyhledávaných hesel. Například uživatel bude chtít hledat překlad hesla pouze v oblasti fyziky. Ukázka uložení takové informace ve formátu LMF u hesla "hmotnost":

```
<LexicalEntry id="00000001">
  <feat att="partOfSpeech" val="commonNoun"/>
  <Lemma>
    <feat att="writtenForm" val="hmotnost"/>
  </Lemma>
  <Sense>
    <Equivalent>
      <feat att='domain' val='fyz.'/>
      <feat att="writtenForm" val="weight"/>
    </Equivalent>
  </Sense>
</LexicalEntry>
```

3.6 Abecední uspořádání

Jak tištěné, tak samozřejmě také elektronické slovníky by měly být abecedně uspořádány. Tato vlastnost výrazně zefektivňuje práci s daty v rozsáhlých souborech. K tomu se využívá abecední uspořádání textů, založené na státní normě příslušného jazyka. Pořadí znaků neboli písmen nemá ve většině případů státních norem žádný logický význam, ale jedná se o zažitý zvyk. I z tohoto důvodu se pravidla pro abecední uspořádání ve státních normách často liší. Nejčastějšími případy jsou písmena s diakritikou. Samotná kontrola abecedního řazení je založena na principu lexikografického uspořádání [11], kde se porovnávají řazené řetězce po znacích ve směru čtení, přičemž při prvním nalezeném rozdílu je rozhodnuto o uspořádání dvou hesel.

Vstupy kontroly abecedního uspořádání hesel jsou pouze slovník a jazyk slovníku. Jazyk z důvodu upřesnění použité státní normy, která stanovuje zásady abecedního zpracování. Výstupem je soubor s hesly, která nesplňují podmínky uvádějící příslušnou státní normou.

3.7 Zpětný překlad

Asi nejpodstatnější implementovanou částí systému je kontrola zpětného překladu hesel slovníků. Zde je kontrola rozdělena do dvou možností spuštění. V prvním případě je k danému slovníku zadáno také heslo, pro které je kontrola prováděna. To znamená, že u překladů vstupního hesla je zjišťována možnost přeložení zpět na původní heslo. Výstupem jsou zde překlady původního hesla s jejich překlady. Případy, u kterých se zpětný překlad povedl, jsou zvýrazněny (viz tabulka 3.2). V druhém případě lze tuto kontrolu spustit pro všechna hesla ve slovníku. Výstupem jsou potom všechna hesla, u kterých tato podmínka neplatí (viz tabulka 3.1). Tato výstupní data jsou rozdělena do tří skupin:

- Překlady původního hesla ve slovník jsou, ale zpět přeložit nejdou.

- Překlady původního hesla ve slovníku nejsou.
- Překladem původního hesla je fráze, která zpět přeložit nejde.

cesta	way
vytvořit	produce
nárazový	shock
zadák	defender
brána	entrance

Tabulka 3.1: Příklad výstupu kontroly zpětného překladu pro všechna slova. Z tohoto příkladu vyplývá, že heslo way nelze přeložit na heslo cesta, heslo produce na vytvořit atd.

path	pěšina **cesta** stezka
road	silnice **cesta** chodba vydat se na cestu
way	způsob ohled směr

Tabulka 3.2: Příklad výstupu kontroly zpětného překladu pro slovo **cesta**.

3.8 Četnost hesel

Tato funkce systému by měla vývojáři zpřístupnit četnosti hesel z množiny zadaných slovníků, které striktně nemusí být ve stejném jazyce, ale pro odlišné jazyky by pravděpodobně funkce postrádala smysl. Tedy například uvidí, která hesla ve kterých slovnících chybí. Operace pracuje pro dva a více vstupních slovníků. Pro každý slovník je generován výstup, obsahující hesla, která se nenachází v daném slovníku, ale jsou dostupná v jednom, nebo více slovnících ze zadané množiny. Ke všem vstupním slovníkům je také vedena statistika, která obsahuje celkový počet chybějících hesel a počty hesel, rozdělených do skupiny, na základě jejich četnosti. Příklad pro upřesnění. Mějme vstupní množinu slovníků – `czen, czge, czfr`. Výstup například pro slovník `czen` by potom vypadal následovně:

```
fena          czge czfr
násypový      czge czfr
nožík         czfr
Vltava        czge
```

Kapitola 4

Implementace systému

Pro implementaci systému byl vybrán programovací jazyk Python (viz kapitola 2.5), konkrétně verze 3.2.3, jen v dále zmíněných výjimečných případech byla využita verze 2.7.3. Hlavním důvodem tohoto výběru je velká podpora práce se soubory a textem. Také oproti ostatním programovacím jazykům převládá jeho syntaktická a sémantická jednoduchost. Celý systém byl implementován, testován a je dostupný na školním serveru Fakulty informačních technologií VUT v Brně `minerva1`.

4.1 Jádro systému

Kořenovým skriptem tohoto systému je `dict_chk.py`. Jeho jediným a nutným argumentem je konfigurační soubor, který obsahuje typ kontroly, vstupní slovníky a v případě nutnosti doplňující informace jako jazyk slovníků, nebo konkrétní heslo pro kontrolu. Pro většinu operací se tedy struktura tohoto konfiguračního souboru mírně liší. Příklady spuštění všech kontrol jsou uvedeny v souboru `README`, dostupném v adresáři této práce. Ve skriptu `dict_chk.py` je soubor zpracován, vstupní slovníky jsou uloženy do struktury `dictionary`[1], která je dále předána jako parametr řídící funkci kontroly, nadefinovaná v konfiguračním souboru. Jediná práce hlavního skriptu `dict_chk.py` je tedy spuštění požadované operace se zadaným slovníkem, či slovníky.

Kontroly jsou implementovány tak, že vstupem bývá jeden elektronický slovník, který je dále testován. Pro usnadnění a zefektivnění práce s tímto systémem byly vytvořeny skripty `makefile.sh` a `make_bt.sh`. `Makefile.sh` vyžaduje na vstupu adresář, ze kterého zpracuje elektronické slovníky a pro každý z nich spustí veškeré dostupné kontroly kromě kontroly zpětného překladu, kde je potřeba slovníky před začátkem testování spárovat. Ve skriptu `makefile.sh` tedy dochází k vytváření konfiguračních souborů pro každou kontrolu zvlášť. Následně je s tímto konfiguračním souborem spuštěn řídící skript `dict_chk.py`. Výjimkou je zde kontrola využívající externí modul morfologického analyzátoru `pylibma`, tedy dostupné rozhraní knihovny `libma` pro programovací jazyk Python. K úspěšnému použití je zapotřebí nastavit proměnné `PYTHONPATH` a `PATH`. Proto je tato kontrola spouštěna skriptem implementovaným v jazyce Shell, kde jsou tyto proměnné nejprve nastaveny, a poté je spuštěn skript v jazyce Python. Kontrola zpětného překladu vyžaduje na vstupu kompletní slovník (např. czen a encz), je tedy potřeba explicitně části slovníků spárovat. Tuto práci provede uživatel ve skriptu `make_bt.sh`, kde je nachystána struktura, která po doplnění slovníků vytváří konfigurační soubory a spouští řídící skript `dict_chk.py`.

[1] Viz `http://www.tutorialspoint.com/python/python_dictionary.htm`

4.2 Zpracování vstupních dat

Vstupní data, tedy elektronické slovníky psané ve značkovacím jazyce XML, které dodržují formát LMF, jsou v jazyce Python zpracovávány modulem SAX. Tento modul dostal přednost před objektovým modelem DOM především proto, že model DOM analyzuje a načítá nejprve celý soubor do paměti a až potom vytváří stromovou strukturu, ze které je možné s daty pracovat. Toto by u nerozsáhlých souborů nebyl problém, ale v našem případě slovníky v LMF formátu mají průměrnou velikost okolo 100 MB. Využití modelu DOM by tedy bylo velmi neefektivní a časově náročné. Upřednostnění před knihovnou jazyka Python ElementTree je zejména proto, že například element `Equivalent` nebývá vždy v XML struktuře stejně zanořený. Je přímým potomkem elementu `Sense`, nebo elementu `Example`. Pro zpracování takovýchto situací je jednodušší na implementaci a v praxi rychlejší model SAX. Jak již bylo zmíněno, SAX je událostmi řízený. Při zpracování vstupních dat pracujeme s událostmi vyvolanými při počátečních a koncových elementech. To znamená, že modul zpracovává XML soubor shora dolů a při každém počátečním elementu je vyvolána událost pro počáteční element a taktéž pro elementy koncové.

Takový přístup zpracování nám dovoluje ze vstupních slovníků získat pouze taková data, se kterými bude spuštěná kontrola pracovat. Výstupem zpracování je potom datová struktura jazyku Python `dictionary`, která je naplněná, jak již bylo zmíněno, jen potřebnými informacemi. Pro tuto práci máme implementovány tři funkce, de facto implementace modulu SAX. Těmi jsou `trans(checked_word,in_dict)`, `trans_back(translations,in_dict)` a `trans_all(in_dict)`. Všechny tyto funkce mají víceméně stejný účel a upřesnění jejich vlastností je v následujících kapitolách.

4.3 Kontrola formátu LMF

Při kontrole formátu LMF u vstupních elektronický slovníků je také, jako v případě kontroly pomocí programu GNU Aspell, využit modul `subprocess`. Tímto modulem je spouštěna další z použitých externích komponent. Testování slovníku s validačním modelem je spouštěno následujícím příkazem:

```
xmllint --noout --relaxng lmf_model_rng.xml + in_dict
```

Z toho `in_dict` je vstupní slovník a `lmf_model_rng.xml` je soubor, ve kterém je vytvořený Relax NG model pro validaci. Výstup je generován automaticky touto technologií. Ve výstupním souboru potom najdeme buďto potvrzení validity vstupního slovníku, nebo specifikované chyby, ve kterých se slovník od modelu liší.

4.4 Srovnání se slovníkem pro kontrolu pravopisu

Pro kontrolu pravopisu hesel jsou do systému zařazeny dva externí programy. Každý z nich je implementován zcela odlišným způsobem. Program GNU Aspell nenabízí žádné rozhraní pro jazyk Python, jinak tomu je u morfologického analyzátoru, který nabízí knihovnu `libma` s rozhraním `pylibma`.

4.4.1 GNU Aspell

Vstupem této kontroly je pouze jedna část slovníku, která musí být napsána v podporovaném jazyce programem GNU Aspell. Každý z těchto jazyků má pevně danou sadu

povolených hodnot, do kterých patří například zkratky, používané v jednotlivých zemích. Tuto sadu hodnot lze snadno upravovat. Hesla ke kontrole ze vstupního slovníku získáme již zmíněnou funkcí `trans_all(in_dict)`. V návratové struktuře nám funkce zpřístupní také informace ke každému heslu jako je jeho id, slovní druh, doplňující gramatické informace a samozřejmě překlad. Máme tedy hesla, která musíme podrobit kontrole, ta jsou zapsána do dočasného souboru. Důvodem je způsob, tedy příkaz, jakým je program GNU Aspell na hesla aplikován. Jak již bylo zmíněno, GNU Aspell nám nenabízí žádné rozhraní pro jazyk Python, proto je tento program spouštěn za pomocí importovaného modulu `subprocess`. Tento modul nám dovoluje spouštět program GNU Aspell v systému unixové platformy. U tohoto příkazu je `tmp_in.txt` dočasný soubor s hesly pro kontrolu, `language` je zkratka jazyka vstupního slovníku a `tmp_out.txt` je dočasný soubor se slovy, která testem neprošla:

```
cat tmp_in.txt | aspell list -l language > tmp_out.txt"
```

Ze souboru `tmp_out.txt` jsou nevalidní slova zpracována. Z těch jsou následovně odebrána slova, která jsou obsažena v sadě povolených hodnot. U souboru `tmp_in.txt` mluvíme o heslech a u souboru `tmp_out.txt` o slovech. Důvod je takový, že vstupní heslo je například "daily dozen", ale programem GNU Aspell neprojde pouze slovo "dozen". Výstupem programu tedy není celé heslo, ale pouze samotné slovo, které testem neprošlo. Ke každému takovému slovu je zpětně dohledáno původní heslo, samozřejmě i s informacemi o něm a vytisknuto do výstupního souboru celé této kontroly. Ne všechna hesla však mají ve slovníku překlad, v takových případech jsou vytisknuta samotná do speciálního výstupního souboru.

Generován je také soubor statistiky, obsahující počet všech kontrolovaných hesel, nekorektních hesel, jejich procentuální zastoupení a celkový čas průběhu kontroly.

4.4.2 Libma

Tato kontrola je jako jediná řízena ze skriptu jazyka Shell, tím je `libma_check.py`. Důvod je jediný, nastavení proměnných `PYTHONPATH` a `PATH`. Jinak by využití rozhraní knihovny `libma` nebylo možné. Poté je v tomto skriptu volán už jen skript jazyka Python `libma_check.py`, ve kterém je samotná implementace kontroly.

Vstupem je opět pouze jedna část slovníku, ale v tomto případě vždy v jazyce českém. Postup je zde velice podobný předešlé kontrole programem GNU Aspell. Získání hesel z výstupu funkce `trans_all(in_dict)`, testování všech hesel a následné filtrování sadou povolených hodnot pro český jazyk. Dostupné rozhraní knihovny `pylibma` však pracuje v kódování `ISO 8859-2`. Vstupní XML slovníky ve formátu LMF jsou v kódování `UTF-8` a tedy také i implementace zpracování modulem SAX. Tudíž je potřeba každé heslo před kontrolou překódovat, aby mohlo být dostupným rozhraním korektně otestováno. Pokud je heslo označeno jako chybné a musí být vytištěno do výstupního souboru společně s veškerými informacemi dostupnými v navrácené struktuře funkce zpracování, musí být heslo překódováno zpět na kódování `UTF-8`. Jak soubor statistiky, tak i výstupní soubory jsou strukturou identické souborům kontroly programem GNU Aspell.

4.5 Atributy slovníků

Veškeré operace nad atributy jsou implementovány ve skriptu `remake_atts.py`. Ve skriptu je funkce `main`, ze která je celá činnost řízena. V té je nejprve volána funkce `get_atts`, která prochází celý vstupní slovník po řádcích. V případě, že nalezne řádek s elementem `feat` a atributem zadaným uživatelem, u kterého se mají jeho hodnoty kontrolovat, tento

element zkopíruje a pošle funkci `check_get_atts`. Příklad takového elementu:

```
<feat att='domain' val='tech., fyz.'/>
```

Pokud je taková informace uvedena u hesla ve slovníku, znamená to, že heslo patří do domény technika a fyzika.

Ve funkci `check_get_atts` je nejprve implementován seznam povolených hodnot formou pole, kde indexem je konkrétní zkratka a hodnotou je význam zkratky. Dále je zjišťováno, kolik je u daného atributu uvedeno hodnot, v našem příkladu jsou to hodnoty dvě (tech. a fyz.). Tyto hodnoty jsou rozděleny a dále se pracuje s každou zvlášť. Počet hodnot, se kterými dokáže systém pracovat, není omezen. Každá hodnota je testována seznamem povolených hodnot. V případě nevalidní hodnoty je celý element vytisknut do výstupního souboru pro návrh na změny v tomto slovníku. Pokud hodnoty kontrolou projdou, je atribut `feat` rozdělen na dva. Například:

```
<feat att='domain' val='tech.'/>
<feat att='domain' val='fyz.'/>
```

Funkci lze také s určitými parametry spustit takovým způsobem, že bude zkratky převádět na hodnoty. To by vypadalo následovně:

```
<feat att='domain' val='technika'/>
<feat att='domain' val='fyzika'/>
```

Tyto změny jsou vráceny zpět do funkce `get_atts`, která původní elementy `feat` nahrazuje těmi upravenými. Vstupní slovník se nemění. Nový opravený slovník je prozatím vygenerován do dočasného souboru, který je dále zpracován funkcí `repair`, jejíž úkol je rozdělit překlady hesel na samostatné atributy. Podobně jako ve funkci `get_atts` je zde hledán konkrétní element. V tomto případě je to element `Equivalent`. Rozdíl je ale v tom, že tento element nevlastní žádné atributy, ba naopak jeho součástí jsou elementy podřazené. Funkci `check_trans_atts`, která elementy `Equivalent` upravuje, je potom zaslána část slovníku, například:

```
<Equivalent>
 <feat att='language' val='cz'/>
 <feat att='writtenForm' val='stroj, automat'/>
</Equivalent>
```

Funkce `check_trans_atts` hodnoty rozdělí a opět vrací zpět funkci `repair`, která v tomto případě už negeneruje do dočasného souboru, ale do výstupního opraveného slovníku. Rozdělené hodnoty jsou vráceny jako samostatné elementy `Equivalent`. U našeho příkladu by to vypadalo takto:

```
<Equivalent>
 <feat att='language' val='cz'/>
 <feat att='writtenForm' val='stroj'/>
</Equivalent>

<Equivalent>
 <feat att='language' val='cz'/>
 <feat att='writtenForm' val='automat'/>
</Equivalent>
```

Dělení hodnot u elementu `Equivalent` je oproti dělení hodnot u elementu `feat` složitější v tom, že hodnoty lze dělit na základě více znaků. Těmi jsou čárka, lomítko, slovo v závorkách a svislítko. Jedna hodnota tudíž může být dělena podle více pravidel (viz tabulka 4.1). Ne vždy je však jasné, jak by hodnota podle určitého znaku měla být rozdělena. Tento případ je vidět u čtvrté a páté hodnoty v tabulce 4.1, kde v prvním případě znak "/" rozděluje pouze slova, tak v druhém případě dělí sousloví. Ke správnému dělení těchto případů by bylo potřeba sémantického analyzátoru, proto tento systém dělí jen hodnoty, u kterých je to zřejmé. Tyto nejasnosti se však týkají pouze hodnot, ve kterých se vyskytuje dělící znak lomítko.

No a (co)?, A co má být?	No a? No a co? A co má být?
Je (ti) to jasné?, Chápeš?	Je to jasné? Je ti to jasné? Chápeš?
způsobit/ přivodit/ způsobovat infarkt	způsobit infarkt přivodit infarkt způsobovat infarkt
zbloudilá střela/ kulka	zbloudilá střela zbloudilá kulka
zbloudilá střela/ zbloudilá kulka	zbloudilá střela zbloudilá kulka

Tabulka 4.1: Příklady možných hodnot elementu `Equivalent` a jejich následné dělení.

4.6 Abecední uspořádání

Potřebnými parametry této kontroly jsou jedna část elektronického slovníku a jazyk pro určení státní normy, na základě které budou hesla testována. Stavebním kamenem této operace je funkce `alphabet(in_dict,language)`, která je taktéž jako funkce ke zpracování slovníků založena na modulu SAX. Tato funkce prochází hesla tak, jak jdou ve slovníku za sebou a pro každé dvě sousední je volána funkce `decide(word1,word2,language)`. Zde jsou slova porovnávána podle normy určené parametrem `language`. Kontrola je dostupná pro státní normu jazyka českého, anglického, ruského, francouzského a německého. Implementačně nejnáročnější je jednoznačně norma jazyka českého [14], definující dvě skupiny písmen s diakritikou, taková, která mají stejnou váhu jako bez diakritiky, a taková, která mají váhu nižší. Čeština je také světově výjimečná písmenem "ch", které je složeno ze dvou samostatných znaků a v abecedním uspořádání má specifickou hodnotu. Návratová hodnota funkce `decide(word1,word2,language)` je `1`, v případě, že hesla jsou uspořádána správně a `2` pokud ne. Správné uspořádání znamená, že parametr `word1` patří abecedně před parametr `word2`. Na vstup funkce jsou hesla přivedena v takovém pořadí, v jakém jsou ve slovníku. Pokud funkce vrátí hodnotu `2`, jsou hesla uložena do struktury typu `dictionary`, která je vrácena funkcí `alphabet(in_dict,language)`. Tato nesprávně seřazená hesla jsou následně jen vypsána do výstupního souboru v takovém pořadí, v jakém jsou ve slovníku (např. "kulky kulka"). Soubor statistiky je zde stejný jako u téměř všech kontrol – tedy počet všech kontrolovaných hesel, nesprávně seřazených, jejich procentuální zastoupení a čas potřebný k provedení kontroly.

4.7 Zpětný překlad

U kontroly zpětného překladu je jako u jediné operace potřeba přivést na vstup celý párový slovník. Jak již bylo zmíněno, tato kontrola má dvě možnosti spuštění. Při kontrole konkrétního hesla je nejprve volána funkce `trans(checked_word,in_dict)`, která má na vstupu kontrolované heslo a první část slovníku. Funkce vrací strukturu typu `dictionary`. Z této struktury lze dostat všechny dostupné překlady kontrolovaného slova. Tuto strukturu společně s druhou částí slovníku přivedeme na vstup funkce `trans_back(translations,in_dict)`, která do struktury doplní překlady všech překladů kontrolovaného hesla a vrátí ji. Následně se do výstupního souboru vypíší překlady kontrolovaného hesla a k nim jejich překlady. Vyznačeny jsou výsledky, které podmínku splňují, a tudíž je lze zpět na kontrolované heslo přeložit.

Kontrolou zpětného překladu je také možno ověřit tuto podmínku pro všechna hesla slovníku. Zde je pro obě části slovníku (např. czen a encz) volána funkce `trans_all(in_dict)`, která vrátí strukturu obsahující všechna hesla slovníku s jejich překlady. Máme tedy k dispozici dvě struktury s hesly a jejich překlady v opačných jazycích. Samotná kontrola je potom vymyšlena tak, že systém prochází postupně všechna hesla v primární struktuře (např. u slovníku `czen` to bude ta, která obsahuje česká hesla s anglickými překlady) a jejich překlady se pokouší hledat ve struktuře sekundární. Zde kontrola generuje tři typy výstupů (viz také kapitola 3.7). V případě, že překlad v sekundární struktuře nalezen nebyl, znamená to, že takové heslo nelze v opačném slovníku hledat a je tedy zařazeno do výstupu hesel nenalezených. Pokud je toto heslo dále prohlášeno za frázi, je zařazeno do výstupu frází s nesplňující podmínkou zpětného překladu. Heslo je prohlášeno za frázi, pokud obsahuje dvě a více slov, anebo zvratná zájmena. Třetí možný výstup hesla nesplňující podmínku nastane tehdy, pokud překlad původního hesla v sekundární struktuře nalezen byl, ale žádný z jeho překladů se nerovná původnímu heslu.

Generována je statistika s celkovým počtem kontrolovaných hesel, hesel nesplňující podmínku, procentuální zastoupení těchto chybných hesel a celkový čas této provedené kontroly.

4.8 Četnost hesel

Zde je potřeba na vstup přivést dva a více slovníků stejného jazyka. Kontrolovat četnost hesel v jednom slovníku by nebylo dost dobře možné. V řídícím algoritmu je implementováno pole, indexované jmény vstupních slovníků. Každá položka tohoto pole obsahuje strukturu typu `dictionary` naplněnou funkcí `trans_all(in_dict)`, kde je na vstupu slovník, kterým je právě položka pole indexována. V jednom poli tudíž máme pod jmény slovníků všechna jejich hesla. Následně u každého slovníku systém prochází všechna hesla a testuje, zda se konkrétní heslo nachází v ostatních slovnících. Pro každý vstupní slovník je generován výstupní soubor s výsledky a soubor statistiky. Soubor s výsledky obsahuje všechna hesla slovníku, kde jsou u každého hesla uvedena jména slovníků, ve kterých se konkrétní heslo taktéž nachází. Soubor statistiky obsahuje počet hesel slovníku a dále počty hesel rozdělených do skupin podle četnosti. Příklad souboru statistiky pro 5 vstupních slovníků:

```
Words, which miss from 3 dictionaries: 110
Words, which miss from 2 dictionaries: 550
Words, which miss from 1 dictionary: 4619
All words, which miss in dictionary: 5279
```

Kapitola 5

Statistiky a výsledky

V této kapitole jsou uvedeny výsledky, statistiky a příklady výstupů kontrol systému. Každé kontrole náleží tabulka s výsledky, tabulka souhrnných ukazatelů a tabulka s několika příklady, převzatými z vygenerovaných výstupů systému. U výsledků je uvedeno jméno testovaného slovníku, počet hesel slovníku, který je totožný s počtem testovaných hesel, celkový čas kontroly v sekundách, rychlost, která udává počet otestovaných hesel za sekundu, a chybovost, což je procentuální vyjádření chybných hesel z celkového počtu testovaných. V tabulkách souhrnných ukazatelů je uveden průměr, maximum a minimum. Tyto informace jsou vypočítány z hodnot předešlé tabulky. Příklady vygenerovaných výstupů jsou u každé kontroly popsány podle konkrétní problematiky. Závěrem této kapitoly jsou dvě tabulky se statistikou, které srovnávají efektivnost kontrol navzájem mezi sebou.

5.1 Libma

V tabulce 5.1 jsou popsány výsledky srovnávání hesel se slovníkem pro kontrolu pravopisu českých slov a v tabulce 5.2 potom souhrnné ukazatele k těmto výsledkům. Tabulka 5.3 obsahuje příklady z výstupu systému pro tuto kontrolu. Pro tuto statistiku bylo otestováno devět elektronických slovníků. Nejvíce nalezených chyb je ve slovníku česko–anglickém, což je 13,2 % z jeho celkového počtu hesel. Nejlépe je na tom český výkladový slovník, u kterého je chybných hesel pouze 1,3 %. Průměrná chybovost u této kontroly je 6,8 %. Nejčastějším typem hesel, která `libma` vyhodnotí jako chybná, jsou hesla převzatá z cizích jazyků.

slovník	**počet hesel**	**čas (s)**	**rychlost**	**chybovost (%)**
czen-lmf	106 481	25,6	4 259,2	13,2
czfr-lmf	54 042	11,1	4 912,9	4,9
czge-lmf	96 952	22,3	4 406,9	11,5
czru-lmf	50 763	10,7	5 076,3	4,2
czsp-lmf	52 385	12,3	4 365,4	3,4
dict-hosp-cz-en	27 315	4,8	6 828,8	1,7
dict-tech-cz-en	45 168	6,2	7 528,0	12,2
dict-velky-cz-en	81 991	10,7	8 199,1	12,8
dic_cz-lmf	21 850	3,2	7 283,3	1,3

Tabulka 5.1: Výsledky srovnávání se slovníkem pro kontrolu pravopisu českých hesel pomocí knihovny `libma`.

	počet hesel	čas (s)	rychlost	chybovost (%)
průměr	59 660	11,4	5 873,3	6,8
maximum	106 481	25,6	8 199,1	13,2
minimum	21 850	3,2	4 259,2	1,3

Tabulka 5.2: Souhrnné ukazatele srovnávání se slovníkem pro kontrolu pravopisu českých hesel pomocí knihovny `libma`.

heslo	překlad	id	slovní druh	gramatická informace
skalára	scalare	e77185g	noun	ž
izotyp	isotype	e26231g	noun	m
lozívat	use to climb	e36482g	verb	ned.

Tabulka 5.3: Příklady z vygenerovaného výsledku u srovnávání se slovníkem pro kontrolu pravopisu českých hesel pomocí knihovny `libma`. Čerpáno z výsledků pro slovník `czen`.

5.2 GNU Aspell

Výsledky ke srovnání se slovníkem pro kontrolu pravopisu hesel v cizích jazycích jsou v tabulce 5.4 a souhrnné ukazatele v tabulce 5.5. Výstup systému je zde stejný jako u kontroly knihovnou `libma` (viz tabulka 5.6). Testováno bylo dvanáct slovníků ve všech dostupných cizích jazycích. Přes padesát procent nevalidních hesel obsahuje pouze slovník německo-český. Pouhá 3,3 % hesel, která neprošla testem jsou v hospodářském slovníku anglicko-českém. Co se týče průměrné chybovosti je na tom GNU Aspel (18,1 %) oproti knihovně `libma` (6,8 %) podstatně hůř. Důvodem může být fakt, že knihovna `libma` je zaměřena pouze na jazyk český, kdežto programem GNU Aspell lze zkontrolovat téměř každý světový jazyk.

slovník	počet hesel	čas (s)	rychlost	chybovost (%)
encz-lmf	143951	1246,5	115,5	30,3
frcz-lmf	53112	46,5	1142,2	10,0
frsk-lmf	52810	40,9	1291,2	9,6
skfr-lmf	50880	34,1	1492,1	10,9
skru-lmf	47976	31,2	1537,7	10,6
sksp-lmf	49330	35,2	1401,4	10,0
rusk-lmf	49206	36,5	1348,1	11,9
rucz-lmf	52287	51,1	1023,2	12,6
gecz-lmf	151737	59,8	2537,4	59,8
dict-hosp-en-cz	28083	10,9	2576,4	3,3
dict-tech-en-cz	104246	288,2	361,7	16,1
dict-velky-en-cz	101327	604,5	167,6	32,2

Tabulka 5.4: Výsledky srovnávání se slovníkem pro kontrolu pravopisu programem GNU Aspell.

	počet hesel	čas (s)	rychlost	chybovost (%)
průměr	73 745,4	207,1	1 249,5	18,1
maximum	151 737	1 246,5	2 576,4	59,8
minimum	28 083	10,9	115,5	3,3

Tabulka 5.5: Souhrnné ukazatele srovnávání se slovníkem pro kontrolu pravopisu programem GNU Aspell.

heslo	překlad	id	slovní druh	gramatická informace
vústenie	abouchement	e45126g	noun	m
fáro	bagnole	e7682g	noun	m
zoomorfný	zoomorphe	e49853g	adjective	príd.

Tabulka 5.6: Příklady z vygenerovaného výsledku u srovnávání se slovníkem pro kontrolu pravopisu hesel v cizích jazycích pomocí programu GNU Aspell. Čerpáno z výsledků pro slovník `skfr`.

5.3 Atributy slovníků

Tabulka 5.7 popisuje výsledky kontroly atributů k zadanému atributu `domain`. Zde můžeme vidět, jako u jediné kontroly celého systému, bezchybnost některých testovaných slovníků. Tabulka 5.8 uvádí ukazatele k těmto výsledkům. Část výstupního souboru je uvedena v tabulce 5.9. Samozřejmě lze kontrolovat jakýkoliv atribut, který je ve vstupním slovníku dostupný. Testováno bylo třináct slovníků, a to takových, které obsahují ve svém LMF formátu atribut `domain`. Bez něj by tato kontrola byla bezvýznamná. Nejhůře prošel kontrolou slovník francouzsko–slovenský s 14,5 % chybných hesel. Průměrný čas u této kontroly je pouze 8,2 sekund. Tato rychlost je způsobena přístupem ke zpracování vstupního slovníku, kde nebyl použit modul SAX, ale slovník je procházen po řádcích a za běhu kontrolován, popřípadě opravován.

slovník	počet hesel	čas (s)	rychlost	chybovost (%)
czen-lmf	40 230	14,9	2 700,0	0
czfr-lmf	13 950	6,6	2 113,6	0
frcz-lmf	1 860	6,2	300,0	8,3
czge-lmf	34 959	14,0	2 497,1	0,005
gecz-lmf	29 029	15,4	1 885,0	12,6
czru-lmf	12 741	6,8	1 873,7	0
rucz-lmf	6 634	6,9	961,4	0,3
czsp-lmf	12 235	7,4	1 653,4	0
sksp-lmf	11 468	6,0	1 911,3	11,7
skfr-lmf	13 128	5,7	2 303,2	12,1
frsk-lmf	1 790	6,2	288,7	14,5
skru-lmf	11 995	5,3	2 263,2	10,9
rusk-lmf	6 350	6,3	1 007,9	7,7

Tabulka 5.7: Výsledky kontroly atributů k zadanému atributu `domain`.

	počet hesel	čas (s)	rychlost	chybovost (%)
průměr	15 105,3	8,2	1 673,7	6
maximum	40 230	15,4	2 700	14,5
minimum	1 790	5,3	288,7	0

Tabulka 5.8: Souhrnné ukazatele pro kontrolu atributů, k zadanému atributu `domain`.

These values of attributes in the element domain are not included in the rules:
domaines
inf.
subj.
service
des
BeF

Tabulka 5.9: Část výsledku kontroly atributů pro zadaný atribut `domain`. Čerpáno z výsledku pro slovník `frcz`.

5.4 Abecední uspořádání

Výsledky ke kontrole abecedního uspořádání jsou uvedeny v tabulce 5.10. Souhrnné ukazatele k těmto výsledkům nalezneme v tabulce 5.11. Příklad z vygenerovaného výstupu je v tabulce 5.12. Testováno bylo třináct slovníků. Dva z těchto slovníků byly zaměřeny na technická hesla a dva slovníky na hesla hospodářská. Tyto neobecné slovníky, zejména potom jejich části anglicko–české, prošly touto kontrolou prakticky bez chyb. U obou bylo pouze 0,1 % chybných hesel. Nejhůře abecedně uspořádaný je obecný slovník česko–anglický.

slovník	počet hesel	čas (s)	rychlost	chybovost (%)
czen-lmf	106 481	18,0	5 915,6	1,5
czfr-lmf	54 042	11,4	4 740,5	1,6
czge-lmf	96 952	16,4	5 911,7	1,6
czru-lmf	50 763	10,6	4 788,9	1,6
czsp-lmf	52 385	8,8	5 952,8	1,4
encz-lmf	143 951	19,2	7 497,5	13,3
dict-hosp-cz-en	27 315	3,0	9 105,0	10,0
dict-hosp-en-cz	28 083	3,1	9 059,1	0,1
dict-tech-cz-en	45 168	5,2	8 686,2	12,3
dict-tech-en-cz	104 246	8,0	13 030,8	0,1
dict-velky-cz-en	81 991	8,2	9 998,9	14,0
dict-velky-en-cz	101 327	9,7	10 446,1	0,2
dic_cz-lmf	21 850	23,4	933,8	0,6

Tabulka 5.10: Výsledky kontroly abecedního uspořádání.

	počet hesel	čas (s)	rychlost	chybovost (%)
průměr	70 350	11,2	7 389,75	4,4
maximum	143 951	19,2	13 030,8	14,0
minimum	27 315	3,0	933,8	0,1

Tabulka 5.11: Souhrnné ukazatele pro kontrolu abecedního uspořádání.

špagety	špagetový
orchidej	orchidea
zmiznout	zmizík
labilnost	labilně
doplňky	doplněk

Tabulka 5.12: Příklady z výsledku pro slovník `czfr` u kontroly abecedního uspořádání.

5.5 Zpětný překlad

Tato kontrola obsahuje dvacet výsledků, což je ze všech kontrol nejvíce. Důvodem je, že pro každý kompletní slovník byla kontrola spuštěna dvakrát. Například nejprve pro česko–německou část slovníku a poté pro německo–českou. Tyto výsledky jsou vedeny v tabulce 5.14, souhrnné ukazatele v tabulce 5.13. Při každém spuštění jsou vygenerovány tři výstupní soubory, příklady z každého z nich jsou v tabulce 5.15. U této kontroly mají až na výjimky všechny testované slovníky vysoké procento chybovosti. U nejhoršího slovníku, slovensko–ruského je to 96,7 %. Takové výsledky mohou být způsobeny například častým překládáním hesel na fráze či sousloví.

	počet hesel	čas (s)	rychlost	chybovost (%)
průměr	70 497,6	18,5	4 107,8	77,3
maximum	151 737	40,2	9 225,3	96,7
minimum	27 315	5,1	2 648,7	39,9

Tabulka 5.13: Souhrnné ukazatele pro kontrolu zpětného překladu.

slovník	počet hesel	čas (s)	rychlost	chybovost (%)
czen-lmf	106 481	40,2	2 648,7	79,4
czfr-lmf	54 042	15,0	3 602,8	80,3
frcz-lmf	53 112	15,1	3 517,3	73,0
czge-lmf	96 952	35,0	2 770,1	92,8
gecz-lmf	151 737	33,3	4 556,6	95,2
czru-lmf	50 763	15,0	3 384,2	96,3
rucz-lmf	52 287	14,8	3 532,9	95,9
czsp-lmf	52 385	16,5	3 174,8	81,0
spcz-lmf	59 240	16,6	3 568,6	75,3
encz-lmf	143 951	37,8	3 808,2	76,7
skfr-lmf	50 880	14,3	3 558,1	79,1
frsk-lmf	52 810	14,2	3 719,0	73,5
skru-lmf	47 976	13,4	3 580,3	96,7
rusk-lmf	49 206	15,4	3 195,2	95,1
dict-hosp-cz-en	27 315	5,1	5 355,8	39,9
dict-hosp-en-cz	28 083	5,1	5 506,4	41,2
dict-tech-cz-en	45 168	11,2	4 032,8	62,4
dict-tech-en-cz	104 246	11,3	9 225,3	83,0
dict-velky-cz-en	81 991	24,2	3 388,1	64,1
dict-velky-en-cz	101 327	16,8	6 031,3	71,7

Tabulka 5.14: Výsledky kontroly zpětného překladu.

překlad nelze přeložit		překlad není ve slovníku		frázi nelze přeložit	
pigment	barvivo	toss	pohození	caner	vyplétač židlí
pigment	zabarvit	off-key	neladící	chaussure	obuv a ponožky
bundle	zmizík	off-key	falešně	cockeye	šilhavé oko
bundle	svazek	weakish	slabší	rebel	vzepřít se
bundle	nacpat	climb	vylezení	amid	uprostřed čeho

Tabulka 5.15: Příklady z vygenerovaných výsledků pro slovník `encz` u kontroly zpětného překladu.

5.6 Souhrnná statistika

Pro celkové srovnání získaných statistik jsou kontroly rozděleny do dvou tabulek, ve kterých jsou seřazeny od nejefektivnější. V první tabulce 5.16 je zohledněn průměrný počet zkontrolovaných hesel za jednu sekundu. V této statistice vítězí kontrola abecedního uspořádání, která v průměru zvládne za jednu sekundu otestovat téměř sedm a půl tisíc hesel. Nejhorší je zde srovnávání se slovníkem pro kontrolu pravopisu programem GNU Aspell, které za sekundu otestuje průměrně pouze necelých 1 250 hesel. Jako pozitivní se jeví fakt, že všechny kontroly zvládnou v průměru otestovat přes tisíc hesel za sekundu.

V tabulce 5.17 jsou potom kontroly seřazeny na základě procentuální hodnoty, která udává, kolik hesel v průměru daná kontrola vyhodnotí jako chybná. U zpětného překladu testem neprojde více než tři čtvrtiny hesel. Nejméně chyb v testovaných elektronických slovnících je v abecedním uspořádání.

abecední uspořádání	7 389,7
libma	5 873,3
zpětný překlad	4 107,8
kontrola atributů	1 673,7
GNU Aspell	1 249,5

Tabulka 5.16: Průměrný počet zkontrolovaných hesel za jednu sekundu u daných kontrol.

abecední uspořádání	4,4 %
kontrola atributů	6 %
libma	6,7 %
GNU Aspell	18,1 %
zpětný překlad	77,2 %

Tabulka 5.17: Tabulka procentuálních hodnot, které udávají, kolik hesel v průměru daná kontrola vyhodnotí jako chybná.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo vytvořit informační systém pro kontrolu překladových a výkladových elektronických slovníků. Implementováno bylo celkem sedm typů kontrol a jedna oprava hromadně měnící obsah slovníků. LMF formát je u vstupních slovníků testován technologií Relax NG, aby v systému nedocházelo k práci s nevalidními elektronickými slovníky. Systém kontroluje abecední uspořádání hesel ve slovnících v pěti jazycích, přičemž největší důraz je kladen na kontrolu jazyka českého. Srovnání se slovníkem pro kontrolu pravopisu hesel je rozděleno do dvou samostatných funkcí. A to pro jazyk český, kde je využito morfologického analyzátoru, a zvlášť pro jazyky cizí, kde je použit program GNU Aspell. Tímto externím programem jsou konkrétně testována hesla v jazyce anglickém, německém, francouzském, ruském a slovenském. Dále je systémem počítána četnost hesel v množině vstupních slovníků. Tato operace není přímo kontrolou, protože výstupem nejsou nalezené ani opravené chyby, ale jen četnosti výskytů hesel. Využití je tedy bráno spíše jako návrh, která hesla by do kterých slovníků měla být doplněna. Kontrola zpětného překladu testuje buďto slovník celý, nebo jen jedno konkrétní heslo, které je zadáno uživatelem. U konkrétního slova je výstup systému mnohem detailnější, jsou zde vypsány všechny překlady hesla a také překlady překladů, které jsou vyznačeny, pokud podmínku zpětného překladu splňují. Zatímco výstupem kontroly celých slovníku je seznam hesel, která podmínku nesplňují. Práce s atributy slovníků má několik úkolů. Na základě předem daných znaků rozděluje hodnoty atributů do atributů samostatných, se kterými jsou generovány opravené slovníky. Dále jsou konkrétní atributy, jejichž jméno je zadáno uživatelem, testovány na povolené hodnoty neboli zkratky, které jsou definovány seznamem k tomuto určeným. Výstupem je návrh na změny hodnot atributů. Poslední funkcí v této oblasti je možnost vygenerovat slovník s převedenými zkratkami na jejich význam.

Z pohledu dalšího vývoje by nejzajímavějším prvkem mohlo být grafické uživatelské rozhraní, které by značně usnadnilo práci s tímto systémem. Zejména při spouštění kontrol a nastavování vstupních parametrů. Systém je dále možno doplnit o další kontroly týkající se elektronických slovníků. Určitý vývoj by také mohl proběhnout v rámci optimalizace jednotlivých kontrol, kde například srovnání se slovníkem pro kontrolu pravopisu hesel v cizích jazycích pomocí programu GNU Aspell je časově velmi náročná a s efektivností ostatních kontrol ji nelze srovnávat.

Literatura

[1] World Wide Web Consortium (W3C) [online]. http://www.w3.org, 1994-06-07 [cit. 2014-04-05].

[2] Welcome to Python.org [online]. https://www.python.org, 1995-03-27 [cit. 2014-04-12].

[3] Linux Shell Scripting Tutorial [online]. http://www.freeos.com/guides/lsst, 1998-11-08 [cit. 2014-04-15].

[4] RELAX NG home page [online]. http://relaxng.org, 1999-03-28 [cit. 2014-04-08].

[5] The ElementTree XML API [online]. https://docs.python.org/2/library/xml.etree.elementtree.html, 1999-03-28 [cit. 2014-05-06].

[6] XML DOM Tutorial [online]. http://www.w3schools.com/Dom, 1999-05-28 [cit. 2014-04-03].

[7] World Wide Web Consortium (W3C) [online]. http://www.w3schools.com/dtd, 2000-03-21 [cit. 2014-04-08].

[8] XPath Tutorial [online]. http://www.w3schools.com/XPath, 2000-03-21 [cit. 2014-04-10].

[9] XML Schema Tutorial [online]. http://www.w3schools.com/Schema, 2000-03-21 [cit. 2014-05-06].

[10] GNU Aspell [online]. http://aspell.net, 2001-01-12 [cit. 2014-04-12].

[11] Lexikografické uspořádání [online]. http://cs.wikipedia.org/wiki/Lexikografické_uspořádání, 2001-01-13 [cit. 2014-04-15].

[12] SAX [online]. http://www.saxproject.org, 2001-10-25 [cit. 2014-04-06].

[13] Kevin Atkinson [online]. http://www.kevina.org, 2012-06-28 [cit. 2014-04-12].

[14] Abecední řazení v češtině [online]. http://www.pistorius.cz/PUMA/Abecední řazení.pdf, [cit. 2014-04-03].

[15] Lexical Markup Framework (LMF) [online]. http://www.lexicalmarkupframework.org, [cit. 2014-04-03].

[16] Schematron [online]. http://www.schematron.com, [cit. 2014-04-10].

[17] Morfologický slovník a morfologický analyzátor pro češtinu [online]. `http://knot.fit.vutbr.cz/wiki/index.php`, [cit. 2014-04-12].

[18] Harold, E.; Means, W.: *XML v kostce*. Computer Press, vyd. 1. vydání, 2002 [cit. 2014-04-03], iSBN 80-7226-712-4.

[19] Young, M. J.; Thiemel, A.: *XML krok za krokem*. Computer Press, vyd. 2. vydání, 2006 [cit. 2014-04-03], iSBN 80-251-1070-2.

Dodatek A

Obsah CD

Přiložené CD obsahuje:

- **dict_chk2014_xsolan01/** – Kořenový adresář projektu, který obsahuje veškeré zdrojové soubory systému a adresáře, do kterých jsou generovány výsledky a statistiky.
- **dict_chk2014_xsolan01_text/** – Adresář obsahuje zdrojové soubory technické zprávy.
- **dict_chk2014_xsolan01.pdf** – Technická zpráva v elektronické podobě.
- **README** – Dokument obsahuje popis zdrojových souborů a návod k práci se systémem.